基于区块链的轻量级匿名评审协议

李 超[1,2], 王 健[1,2], 刘吉强[1,2]

[1] 北京交通大学智能交通数据安全与隐私保护技术北京市重点实验室 北京 中国 100044
[2] 北京交通大学计算与信息技术学院 北京 中国 100044

摘要 同行评审的重要价值一直被学术界广泛认可, 然而其过程的不透明广受诟病。近年来, 区块链技术的快速发展正在迅速推动以太坊等开放式智能合约平台的成熟, 为开发去中心化的评审系统奠定了坚实基础。然而, 目前去中心化的评审协议面临两个有挑战性的问题。首先, 由于区块链记录的信息是透明公开的, 若评审方的身份在评审结果产生前被公开, 会导致匿名性难以保障, 不利于维护评审过程的公平性。其次, 由于智能合约中函数的每一次调用都要花费一定量链上资源, 执行包含 n 位评审方的协议需花费 O(n) 链上资源, 导致可扩展性难以保障, 协议难以应用到实际场景。本文提出一种基于区块链的轻量级匿名审稿协议(Blockchain-based Lightweight Anonymous Review, BLAR), 旨在解决去中心化评审协议的匿名性和可扩展性两个关键问题。BLAR 协议不需要在评审结果展示前在区块链上存储任意可能导致指派信息泄露的信息, 包括但不限于被选中评审方的账户地址或其哈希值, 从而使攻击者无法确定性地找出对应某投稿的评审方。同时, BLAR 协议不依赖区块链进行存储与计算, 而是仅利用区块链进行验证和可信性保证, 从而在最小程度造成区块链链上负担的情况下, 保证数据汇聚与处理的可信性。我们证明, 只要主办方和评审方存在至少一位诚实参与者, BLAR 协议一定能执行完毕, 且一定能产生符合评审者打分的正确结果。我们还证明, 当全部协议参与方具备理性时, 执行 BLAR 协议全过程的成本为 O(1) , 独立于参与方数量的规模。我们在以太坊官方测试网络上实现了 BLAR 协议, 并进行了实验评估。结果表明, 无论协议参与方的规模大小, BLAR 协议都可以将匿名评审的执行成本降低到 1 美元以下, 远低于现有工作中方案的执行成本, 具备了实用性。

关键词 区块链; 智能合约; 匿名评审系统; 可扩展性; 活性
中图法分类号 TP309.7 **DOI 号** 10.19363/J.cnki.cn10-1380/tn.2022.09.08

Blockchain-based Lightweight Anonymous Review System

LI Chao[1,2], WANG Jian[1,2], LIU Jiqiang[1,2]

[1] Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, Beijing 100044, China
[2] School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China

Abstract The important value of peer review has been widely recognized by the academic community, but the process of peer review has all along been criticized to be lacking transparency. In recent years, the rapid development of blockchain technology is quickly promoting the maturity of open smart contract platforms such as Ethereum, laying a solid foundation for the development of decentralized review systems. However, decentralized review protocols currently face two challenging problems. First, since the information recorded by the blockchain is transparent and open, the identities of the reviewers are disclosed before the review results are produced, which makes anonymity difficult to be guaranteed and is not conducive to maintaining the fairness of the review process. Second, because each invocation of a function in a smart contract costs a certain amount of money, a protocol involving n reviewers could result in O(n) monetary cost, it is thus difficult to guarantee scalability and to apply such a protocol to real scenarios. In this paper, we propose a blockchain-based lightweight anonymous review protocol (BLAR) to solve the two key problems, anonymity and scalability, faced by decentralized review protocols. The BLAR protocol does not need to store any information that may lead to disclosure of assignment information on the blockchain prior to announce of review results, including but not limited to the account address or hash value of the selected reviewers, thus making it impossible for an attacker to identify the reviewers assigned for a manuscript. Also, BLAR protocol does not rely on blockchain for storage and calculation, but only uses blockchain for verification and credibility guarantee, so as to ensure the credibility of data aggregation and processing with the minimum burden on the blockchain chain. We demonstrate that as long as there is at least one honest participant between the host and reviewers, the BLAR protocol cannot be aborted, and the results are guaranteed to be correct. We also show that when all parties are rational, the cost of implementing the entire BLAR protocol could be reduced to only O(1), regardless of the scale of participants. We implemented and experimentally evaluated the BLAR protocol on the official Ethereum test network. Our results demonstrate that BLAR protocol can reduce the cost to less than $1 regardless of the scale of the participants, indicating that BLAR protocol is a practical solution with its cost much lower than the implementation cost of

通讯作者: 刘吉强, 博士, 教授, Email: jqliu@bjtu.edu.cn。
本课题得到国家重点研发计划课题(No. 2020YFB2103802), 北京市自然科学基金(No. M22039)资助。
收稿日期: 2021-08-26; 修改日期: 2022-01-13; 定稿日期: 2022-07-15

existing solutions.

Key words blockchain; smart contract; anonymous review system; scalability; liveness

1 引言

同行评审是学术论文选拔与质量把控的必要步骤[1-2], 不仅关乎投稿人稿件的命运, 也决定学术期刊或会议的发展, 甚至影响整个学科领域的兴衰。通常, 期刊编委或会议主席将一篇投稿分配给复数位评审者, 在一定期限内收集评审者反馈的稿件级别(如拒绝、重投、大修、小修、接收)或分数(如区间[0,5]内的整数), 依此做出是否接收该投稿的进一步决定, 这一流程在实际执行中可能进行多轮, 直到该投稿的命运被最终决定。

尽管同行评审的重要价值一直被学术界广泛认可, 同行评审过程的不透明与结果的不公开却广受诟病。通常, 同行评审是一个保密的过程, 作者不知道评审者的身份, 评审报告的内容与分数也不对外部公开。然而, 近年来, 学术界正逐渐重视同行评审的透明性, 越来越多的机构和学者呼吁在评审完毕后公开评审结果的信息, 包括评审报告的内容与分数, 甚至评审者的身份。2020 年 2 月, 《自然》杂志发表社论[3], 宣布将尝试公开评审意见。《自然》指出, 在面向该刊审稿人的调研中, 超过一半的受访者希望出版社能进一步推进同行评审的透明化, 该刊已有约 3700 名审稿人选择在评审后公开身份, 且该刊超过 80%的论文中至少有一位审稿人选择了公开身份。《自然》宣称, 将进一步推进自身及其 7 种子刊的评审透明化。此外, 近年来, NeurIPS、ICLR 等多个计算机领域顶级国际学术会议在逐步推进评审结果的透明化。2021 年, NeurIPS 宣布将采用 OpenReview 系统[4]来完成整个审稿过程, 并公开评审意见与作者回应。可见, 得益于学术界的共同努力, 同行评审的透明化正得到迅速推进, 从而可能一方面提升评审意见质量, 另一方面遏制评审者的不合规行为, 包括评审者故意对低质量投稿打高分以促使其被接收, 或故意对高质量投稿打低分以阻止其被收录。

尽管 OpenReview 等同行评审系统不断涌现, 这些系统的中心化本质可能导致一系列问题。这些中心化的系统迫使其用户选择相信系统运营方, 导致系统的安全性受制于对单一实体的信任。更重要的是, 即使系统运营方确实是可信的, 评审过程仍可能面临运营方无法控制的安全事故, 例如无法预期的安全漏洞和内部攻击[5-6]。近年来, 起源于比特币[7]的区块链技术取得迅速发展。区块链建立在去中心化的对等网络之上, 交易(transaction)以数据形式由全网节点备份, 交易顺序和内容的一致性和不可篡改性由共识机制保障。区块链开创性地构建了去中心化的信任, 使人们可以选择信任区块链底层密码学技术的可靠性及对等网络中大部分节点的诚实性, 而无需被迫信任单一实体。以比特币为代表的第一代区块链仅支持用户间的转账交易, 缺乏金融领域外的应用空间。2014 年, 以太坊[8]作为首个支持图灵完备智能合约[9]的第二代区块链平台, 使区块链具备可编程性, 极大地拓展了区块链技术的应用场景。截止到 2021 年 8 月, 以太坊的市值已超过 3650 亿美元[10], 基于以太坊开发的去中心化应用(decentralized application, DAPP)已超过 2800 个[11], 覆盖社交、开发、游戏、服务、健康等众多领域。去中心化应用 DAPP 与传统 APP 应用的核心区别在于, DAPP 的后端逻辑以智能合约方式部署在区块链上, 一个 DAPP 可包含多个智能合约, 每个智能合约通常由多个函数组成。每个函数的执行需由用户提交到区块链中的对应交易触发。当区块链网络中的节点接收到用户的交易, 将基于交易内容执行指定智能合约中的指定函数, 函数的执行结果将由全网节点验证, 从而在去中心化环境中保障执行结果的确定性和正确性, 进而舍弃对中心化实体的依赖。以太坊拥有成熟的技术与活跃的社区, 为构建去中心化的评审系统, 并克服中心化评审系统的单点信任缺陷奠定了坚实基础。近期, 文献[12]提出基于以太坊的评审协议 Ants-Review, 该协议将评审过程的管理流程编写为智能合约, 要求投稿方、评审方、(会议或期刊)主办方等协议参与方按照协议规定调用函数, 完成稿件的评审流程。Ants-Review 协议成功实现了评审流程的去中心化, 但该协议面临两个具有挑战性的问题。首先, 现阶段的 Ants-Review 协议无法保障评审的匿名性。具体来说, 投稿的评审人一旦被指派, 其身份将被记录进账本, 目的是对指派进行不可篡改的记录, 以在未来进行审计、奖励和追责。然而, 由于区块链记录的信息是透明公开的, Ants-Review 中评审方的身份在评审结果产生前被公开, 不利于维护评审过程的公平性。其次, Ants-Review 协议不具备可扩展性。具体来说, 该协议要求每位评审方通过调用智能合约中的数个函数完成评审, 然而以太坊中每个函数的调

用都需要花费一定数量链上资源, 这导致包含 n 个评审方的协议执行需花费 O(n) 链上资源, 难以被应用到实际场景。

本文提出一种基于区块链的轻量级匿名审稿协议 BLAR(Blockchain-based Lightweight Anonymous Review), 旨在解决去中心化评审协议的匿名性和可扩展性两个关键问题。BLAR 协议不需要在评审结果展示前在区块链上存储任意可能导致指派信息泄露的信息, 包括但不限于被选中评审方的账户地址或其哈希值, 从而使攻击者无法确定性地找出对应某稿件的评审方。同时, BLAR 协议不依赖区块链进行存储与计算, 而是仅利用区块链进行验证和可信性保证, 从而在最小程度造成区块链链上负担的情况下, 保证数据汇聚与处理的可信性。我们证明, 只要主办方和评审方存在至少一位诚实参与者, BLAR 协议一定能执行完毕, 且一定能产生符合评审者打分的正确结果。我们还证明, 当全部协议参与方具备理性时, 执行 BLAR 协议全过程的成本为 O(l), 独立于参与方数量的规模。我们在以太坊官方测试网络上实现了 BLAR 协议, 并进行了实验评估。结果表明, 无论协议参与方的规模大小, BLAR 协议都可以匿名评审的执行成本降低到 1 美元以下, 远低于 Ants-Review 协议的执行成本, 具备了实用性。

本文余下各章节内容如下: 我们首先在第二节中介绍与本文内容相关的背景知识; 在第三节中, 我们描述本文采用的多个模型, 包括基于区块链的匿名评审模型与安全模型, 并介绍协议设计的目标; 在第四节中, 我们首先介绍一种基础的基于区块链的评审协议, 分析改进该基础协议以支持匿名和轻量化的主要挑战, 从而提出改进后的轻量级匿名评审协议 BLAR; 接着, 我们在第五节中对 BLAR 协议进行安全分析与成本分析, 并在第六节中在以太坊官方测试网络上实现和评估 BLAR 协议, 并与现有工作进行对比; 最后, 我们在第七节对全文工作进行总结。

2 背景知识

在本节中, 我们介绍区块链与智能合约相关的背景知识, 并介绍本文采用关键密码学工具。为了便于表述, 考虑到以太坊[8]是近年来最具代表性的智能合约平台, 拥有最广泛的用户群体与最活跃的社区环境, 本文中智能合约的相关讨论在以太坊环境中进行, 但文中的解决方案也适用于其他智能合约平台。

2.1 区块链与智能合约

在这一小节, 我们基于以太坊背景环境, 介绍与本文相关的区块链与智能合约的基础知识。

2.1.1 外部账户与合约账户

以比特币[7]为代表的多数第一代区块链系统仅存在一种类型的账户, 通常对应一对公私密钥, 其中公钥通常可映射到账户地址, 私钥由用户钱包应用保存在本地, 对用转账交易进行数字签名。

作为第二代区块链系统的典型代表, 以太坊实现了区块链的可编程性, 支持图灵完备的智能合约, 因而存在两种不同类型的账户, 即外部账户(Externally Owned Account, EOA)和合约账户(Contract Account, CA)。其中, 外部账户类似比特币账户, 是由以太坊用户创建, 通过一对公私密钥控制的账户。用户的钱包应用首先生成私钥, 采用椭圆曲线数字签名算法 ECDSA-secp256k1 将私钥映射成公钥, 最后通过 SHA3 生成账户地址。以太坊中的每个智能合约对应一个唯一的合约账户, 合约账户由外部账户以创建智能合约的方式创建, 并由被创建的智能合约的代码控制。合约账户的地址由合约创建者地址, 以及该地址发出的交易数目共同计算得出。

举例来说, 为了与以太坊区块链交互, 用户需要创建外部账户(EOA), 并通过一对密钥来控制它。然后, 用户都可以通过创建的 EOA 发送一类特殊交易来创建智能合约, 被创建的智能合约将自动获得合约账户, 以及 20 字节地址作为该账户唯一标识。

2.1.2 交易类型与交易费用

以比特币为代表的第一代区块链系统普遍仅支持单一类型的交易, 即账户间转账交易。以太坊将可编程的区块链为目标, 设置了更复杂的交易结构和类型, 以支持智能合约的部署和调用。以太坊中交易结构的核心字段如表 1 所示。根据交易接收者地址(to 字段)类型的不同, 以太坊中存在三种类型的交易。

表 1 以太坊交易结构

Table 1 Transaction structure in ethereum

核心字段	描述
from	交易发送者地址
to	交易接收者地址
value	以太币转账数量
data	二进制数据有效负载
v, r, s	交易发送者的数字签名

转账交易: 当 to 字段为 EOA 地址且 value 字段

为非空时, 交易将被以太坊虚拟机识别为转账交易。该类型交易用于将 value 字段指定数额的以太币(以太坊中的密码学货币)从发送方控制的 EOA 转移到接收方控制的 EOA。发送方 EOA 的账户余额必须大于 value 字段的值, 否则交易将失败。

合约创建交易: 当 to 字段为空(0x0)地址且 data 字段为非空时, 交易将被以太坊虚拟机识别为合约创建交易。该类型交易用于创建新的智能合约。具体来说, 用户首先使用高级的面向合约的编程语言(如 Solidity[13])创建的一段智能合约程序, 随后将高级语言的智能合约编译为以太坊虚拟机可识别的低级字节码语言, 并在构建交易的过程中将字节码填入 data 字段。合约创建交易进入区块后, 矿工将分配新的用于该合约对应 CA 的存储空间, 并将 data 字段的字节码存储于该空间内。

函数调用交易: 当 to 字段为 CA 地址且 data 字段为非空时, 交易将被以太坊虚拟机识别为函数调用交易。该类型交易用于调用已创建的智能合约中的函数。具体来说, 在构建交易的过程中, 用户将目标智能合约的 CA 地址填入 to 字段, 并将该合约中拟调用函数的相关信息填入 data 字段, 包括拟调用函数的名称、输入参数的类型和值。此后, 矿工可通过读取 to 字段和 data 字段的内容, 定位用户拟调用的函数所在 CA, 从该 CA 对应存储空间的字节码中获取拟调用函数对应的逻辑代码, 并基于用户提供的输入参数值计算该函数的执行结果, 最终将结果更新到区块链状态中。

综上, 三种类型的交易遵从相同的交易结构, 以太坊虚拟机根据交易中不同字段内容的格式, 自动对交易进行分类, 并采取不同策略根据交易执行结果更新区块链状态。例如, 转账交易通常使发送方与接收方的账户余额发生变动, 合约创建交易通常产生新的智能合约及合约账户 CA, 函数调用交易可改变被调用合约中的变量值, 及改变发送方与接收方的账户余额。

以太坊交易具备可追溯性, 且执行结果难以被篡改。用户在构建交易的最后一步, 基于椭圆曲线数字签名算法, 使用所控制的 EOA 的私钥对交易进行数字签名, 并将签名信息 v、r、s 三元组填入交易体中一并发送。基于该签名信息, 任何人可计算并核实签名者, 即交易发送方 EOA 地址, 从而使交易具备可追溯性。此外, 类似比特币, 以太坊采用工作量证明(Proof of Work, PoW)共识协议[14]。简单地说, 以太坊中数以万计的矿工持续消耗算力计算难题(puzzle), 最先解出答案的矿工有权生成包含交易内容的最新区块, 此后全网基于该区块中交易的执行结果对区块链状态进行同步以保持一致性。攻击者如想篡改交易执行结果, 需具备可与全网矿工竞争的海量算力, 这在实际中很难实现, 因而交易的执行结果普遍被认为是难以篡改的。

值得注意的是, 以太坊中交易的执行并非免费, 需要花费燃气(Gas), 进而产生交易费用。以太坊中任一笔交易首先收取 21000 Gas 的基础费用, 在此基础上, 交易执行过程中的每一条指令都根据其种类收取对应的 Gas 费用。例如, 通过执行 SSTORE 指令将某一变量值从零变为非零会收取 21000 Gas, 通过执行 CALL 指令创建了一个新的账户会收取 25000 Gas。根据交易执行的复杂程度, 一笔交易花费的 Gas 总量可达数百万, 但不可超过单一区块所允许的最大 Gas 量限制。在以太坊中, 为了发送可执行的交易, 用户需要保证作为交易发送方的 EOA 余额足够, 这是因为矿工在执行交易时, 会按照浮动的汇率从发送方 EOA 收取交易费用, 即收取对应交易花费 Gas 总量的以太币。一旦余额不足, 交易将失败, 区块链状态会回滚。Gas 系统对以太坊很重要, 交易费用被用于奖励矿工, 这有助于激励矿工保持诚实, 抑制拒绝服务攻击, 并鼓励提升智能合约编程的高效性。另一方面, Gas 系统对协议设计的可扩展性提出了更高的要求, 在参与者众多的协议中, 即使单一笔交易的费用不高, 一轮协议产生的总交易费用可能非常高。

2.1.3 节点间通讯

同多数区块链系统一样, 以太坊中的节点组成点对点(peer-to-peer, P2P)网络。以太坊社区提出了 Whisper 协议[15], 以支持 P2P 网络中的节点间通讯。在默认情况下, 基于 Whisper 协议发送的消息被广播到整个 P2P 网络, 所有消息必须以对称或非对称的方式加密, 并可被拥有对应密钥的节点解密。例如, 一个节点可以在本地生成一对非对称的密钥, 并将公钥存储到区块链上, 使公钥全网可见, 从而获得其他节点利用该公钥和 Whisper 协议传递的加密消息, 并对该消息解密获悉其中内容。

2.2 本文所用到的密码学工具

在这一小节, 我们介绍本文中协议的设计使用到的几个关键的密码学工具。

2.2.1 默克尔树

默克尔树[16]是一种二叉树, 每一个节点都是其两个子节点的哈希, 最顶层节点是根节点。基于根节点和默克尔证明, 用户可以在 $O(\log n)$ 时间内完成对 n 个叶子节点中任一节点的验证。因此, 默克尔树作

为一种高效的方法, 仅需将根节点哈希值存储在区块链上, 便可验证某特定数值在一大组数值中是否存在, 被广泛应用于区块链领域。例如, 以太坊中的世界状态树、交易树、收据树采用的数据结构均是默克尔树的变体。本文使用了默克尔树的标准概念, 并将从叶节点集合 $\{leaf_i\}_{i=1}^{n}$ 获得根节点哈希值 *root* 的运算表示为 $root \leftarrow \mathbf{M}(\{leaf_i\}_{i=1}^{n})$。

2.2.2 Keccak-256 哈希函数

Keccak-256[17]作为 SHA-3 密码哈希函数竞赛的获胜算法, 被广泛应用在包括以太坊在内的区块链系统中。以太坊 Solidity 语言提供函数 keccak256(…) returns (bytes32), 可直接在智能合约中计算哈希值。本文使用的哈希函数均为 Keccak-256, 哈希运算表示为 $hash \leftarrow \mathbf{H}(\cdot)$。

2.2.3 ECDSA 数字签名

以太坊和比特币等多个主流区块链系统均采用椭圆曲线数字签名算法(Elliptic Curve Digital Signature Algorithm, ECDSA)[18]。ECDSA 签名由两个整数 $\{r,s\}$ 组成, 在此基础上, 以太坊引入额外的恢复标识符 v, 形成三元组签名 $\{v,r,s\}$。以太坊社区提供 JavaScript API 对任意消息 *msg* 进行签名, 签名运算表示为 $vrs \leftarrow \mathbf{S}(\mathbf{H}(msg))$, 其中 $\mathbf{H}(msg)$ 为 *msg* 的 Keccak-256 哈希。此外, 以太坊 Solidity 语言提供全局函数 ecrecover(…) returns (address), 可直接在智能合约中校验签名, 表示为 $signer \leftarrow \mathbf{V}(\mathbf{H}(msg),vrs)$, 输出的 *signer* 为签名者的以太坊地址。

2.2.4 星际文件系统

星际文件系统(InterPlanetary File System, IPFS)是采用基于内容寻址、分布式的、点对点的新型超媒体传输协议的新型分布式文件系统[19]。存储在 IPFS 中的文件对应基于文件内容的不变的永久性链接。该链接关联文件内容的哈希值, 因此文件内容的微小变化将导致完全不同的链接, 而使用同一链接下载的文件内容一定一致。通过将文件的 IPFS 链接存储到区块链中, 不同用户可利用该链接从 IPFS 系统获取内容一致的文件, 这避免了直接将文件存储到区块链中并共享给用户所产生的昂贵费用。为了降低在以太坊区块链上的数据存储成本, 本文采用 IPFS 进行文件共享, 表示为 $link \leftarrow \mathbf{I}(file)$, $file \leftarrow \mathbf{R}(link)$。

3 模型与假设

在本节中, 我们首先描述基于区块链的评审系统模型。在此基础上, 我们介绍本文采用的安全模型。最后, 我们说明本文协议的设计目标。

3.1 评审系统模型

在这一小节, 我们首先对匿名评审全过程进行抽象, 接着描述传统中心化的评审系统模型, 最后提出基于区块链的评审系统模型。

参考学术论文的同行评审场景, 我们将匿名评审全过程抽象为四个阶段, 即论文提交阶段、匿名指派阶段、论文评审阶段、结果公示阶段。匿名评审全过程涉及到三类参与方, 即期刊或会议的主办方(Host,表示为 H)、m 位投稿方(Creater,表示为 $C_{i=1}^{m}$)、n 位评审方(Reviewers,表示为 $R_{j=1}^{n}$)。接下来, 我们首先描述不使用区块链的。

参考 OpenReview[4]等传统中心化的评审系统, 我们抽象出中心化的评审系统模型, 如图 1 所示。在提交阶段, 主办方在期刊或会议网站发布投稿要求(如会议的 Call For Paper), 投稿方向网站提交论文。在指派阶段, 主办方对每篇论文的评审方进行指派, 并邮件通知指派结果。在提交阶段, 评审方向期刊或会议网站提交分数。最后, 在公示阶段, 主办方向每位投稿方邮件通知评审结果。

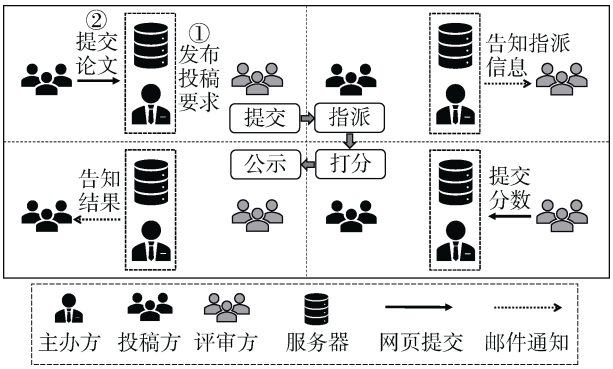

图 1 中心化的评审系统模型

Figure 1 Centralized review system model

参考 Ants-Review[12]等去中心化的评审系统, 结合本文抽象出的四个阶段, 我们提出本文使用的基于区块链的评审系统模型, 如图 2 所示。该模型将图 1 中心化的服务器替换为去中心化的智能合约, 以克服中心化系统的单点瓶颈。有别于图 1 中参与方间的两类交互方式, 图 2 中参与方间信息交互采用被相关研究[20-21]广泛认可的两类模式:

(1) 链上公开模式: 参考 2.1.2.小节介绍的以太坊交易结构及类型, 参与方可将信息(如论文、投稿要求、评审分数等)填入函数调用交易, 通过调用智能合约中的函数, 使该信息被记录进区块链, 并向

全网节点公开。在该模式中, 我们假设区块链系统具备一致性、可用性及不变性。具体来说, 当一个参与方通过交易向区块链提交一条消息, 在有限的时间区间 T 内, 其他参与方可从账本中获得该消息, 内容一致且在未来不可变。

(2) 链下通讯模式: 参考 2.1.3 小节介绍的以太坊节点间通讯协议 Whisper[15], 参与方可利用 Whisper 协议建立点到点的通信信道, 从而向特定的接收方传递信息。我们假设每个协议参与方都可建立与任意其他参与方间的链下信道, 且该信道是可靠的, 即通过该信道传输的信息不会丢失。我们假设链下通讯符合同步模型, 即通过该信道传输信息的时延存在确定的上界 T'。

图 2 中四个阶段具体描述如下:

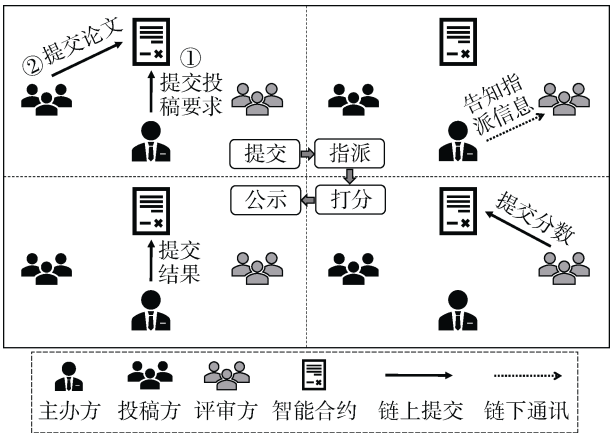

图 2 基于区块链的评审系统模型图

Figure 2 Blockchain-based review system model

(1) 论文提交阶段: 在此阶段中, 主办方首先提交投稿要求到智能合约, 投稿方在时限内提交符合主办方要求的论文到智能合约。

(2) 匿名指派阶段: 在此阶段中, 主办方基于自主定制的规则(如基于声望、基于适配程度等), 建立投稿集合与评审方集合的映射。主办方通过链下信道告知每位评审方所需评审的稿件, 但不向任何第三方透露该指派信息。

(3) 论文打分阶段: 在此阶段中, 评审方按照主办方在指派阶段分配的稿件信息, 对稿件进行打分, 且须在时限内将打分结果提交到智能合约。主办方根据智能合约上的打分结果, 统计各稿件最终分值与排序。

(4) 结果公示阶段: 主办方向智能合约提交评审结果及评审者名单。如对公示结果有异议, 或对主办方、评审方操作行为有异议, 可在此阶段提出质疑。

3.2 安全模型

在这一小节, 我们首先描述攻击者假设, 接着介绍 any-trust 安全模型, 最后讨论攻击者可能采取的攻击行为。

攻击者假设: 参考文献[22-23]中提出的攻击者假设, 协议中参与方的安全性存在四种可能, 即可信参与方、半诚实攻击者、理性攻击者、恶意攻击者。具体来说, 可信参与方诚实地执行协议, 不会违背协议。三类攻击者的区别如下:

(1) 半诚实攻击者: 该类型攻击者的行为遵循协议, 但试图从可用的中间结果获取更多信息;

(2) 恶意攻击者: 该类型攻击者可以采取任何恶意行为, 其行为不顾及自身利益;

(3) 理性攻击者: 该类型攻击者的行为由自身利益驱动。在面临遵循协议还是违法协议的选择时, 理性攻击者会基于自身预期收益进行选择。当遵循协议带来的预期收益更高时, 会选择遵循协议, 只有当违反协议的预期收益更高时, 才会选择违反协议。例如, 如果该类型攻击者能够预期其违反协议(提交错误结果)的效果为 0(错误结果会被纠正)且惩罚大于 0(没收押金), 便不会选择违反协议, 而恶意攻击者在这种情况下仍可能选择违反协议。

近年来, 大量研究认为, 在许多实际情况下, 半诚实攻击者的假设太弱, 而恶意攻击者的假设太强, 因此由个人利益驱动的理性攻击者假设在许多攻击场景中更贴合实际。

Any-trust 安全模型: 参考文献[24-25]中提出的 any-trust 安全模型, 本文设置一条维护协议安全性的底线, 即在主办方 H 和 n 位评审方 $R_{j=1}^{n}$ 这 $(1+n)$ 位参与方中仅存在一位诚实遵循协议的可信参与方, 该可信参与方既可能是主办方 H, 又可能是某位评审方 R_j。换言之, 只要存在至少一位可信参与方, 协议安全性就能得到保障。我们进一步假设, 除这一位可信参与方以外的全部 n 位参与方都既可能是理性攻击者, 又可能是恶意攻击者(由于半诚实攻击者的假设太弱, 我们剔除该假设)。也就是说, 当仅存在一位可信参与方时, 存在两种极端可能:

(1) 最实际 any-trust 安全模型: $(1+n)$ 位参与方包含一位可信参与方以及 n 位理性攻击者, 没有任何参与方会在有损自身利益的情况下进行攻击行为。

(2) 最坏 any-trust 安全模型: $(1+n)$ 位参与方包含一位可信参与方以及 n 位恶意攻击者。

此外, 考虑到多数情况下主办方相对于评审方更缺乏泄露评审信息的动机, 如论文出版方希望获得更优秀的投稿, 我们假设主办方不会在公示期前泄露指派信息, 但评审方可能会出于自身利益在公

示期前泄露指派信息。

攻击行为: 攻击行为指攻击者采取的违背协议内容的行为。在基于区块链的评审系统模型中, 攻击行为仅可能发生在打分阶段和公示阶段, 即三类参与方全部加入协议后。具体来说, 如果主办方没有提交投稿内容, 或投稿方没有提交论文, 或评审方没有接到审稿通知, 则三类参与方间契约尚未达成。在打分阶段, 协议要求评审方提交分数, 评审方可能的攻击行为包括不提交分数或重复提交分数。在公示阶段, 协议要求投稿方提交结果, 投稿方可能的攻击行为包括不提交结果或提交错误结果。我们将在 4.3.5 小节对这四种攻击行为进行更进一步的讨论。

3.3 协议设计目标

本文拟提出安全、可扩展的匿名评审协议, 使协议在最坏 any-trust 安全模型下依然是安全的, 而在最实际 any-trust 安全模型下能做到可扩展。

在安全性方面, 本文要求, 在仅存在一位可信参与方, 其余 n 位参与方均为恶意攻击者的最坏 any-trust 安全模型下, 协议也须是安全的。

定义 1. 安全匿名评审协议. 面向基于区块链的评审系统模型设计的协议满足下述安全特性:

活性(liveness): 只要主办方和评审方存在至少一可信参与方, 协议一定能按四阶段顺序执行完毕, 且一定能产生评审结果;

正确性(correctness): 只要主办方和评审方存在至少一位可信参与方, 产生的评审结果一定是符合诚实评审者打分的正确结果;

匿名性(anonymity): 只要主办方在公示期前不透露指派信息, 评审方无法证明已被主办方选择进入评委会。

在可扩展性方面, 本文要求, 在仅存在一位可信参与方, 其余 n 位参与方均为理性攻击者的最实际 any-trust 安全模型下, 协议是可扩展的。由于比特币、以太坊等区块链的链上写入及操作成本极高, 本文主要关注链上通讯的效率, 旨在降低协议在以太坊环境中执行消耗的 Gas 量。

定义 2. 可扩展匿名评审协议. 当全部协议参与方具备理性时, 面向基于区块链的评审系统模型设计的协议, 在执行全过程的区块链链上 Gas 消耗为 O(1), 即消耗独立于参与方的规模。

4 协议设计

在本节中, 我们首先介绍一种基础的基于区块链的评审协议, 该协议描述了主办方、投稿方、评审方三类协议参与方在提交、指派、评审、公示 4 个阶段的关键交互步骤, 然而不支持匿名, 且链上 Gas 消耗极高。在此基础上, 我们分析改进该基础协议以支持匿名和轻量化的主要挑战, 并提出攻克这些挑战的关键技术。最后, 我们提出改进后的轻量级匿名评审协议 BLAR。

4.1 基础协议

基础协议概览如图 3 所示。该协议的链上逻辑以包含多个函数的智能合约形式存在。参与方可通过函数调用交易执行所需函数(图中实线), 也可通过链下信道传递信息(图中点线)。

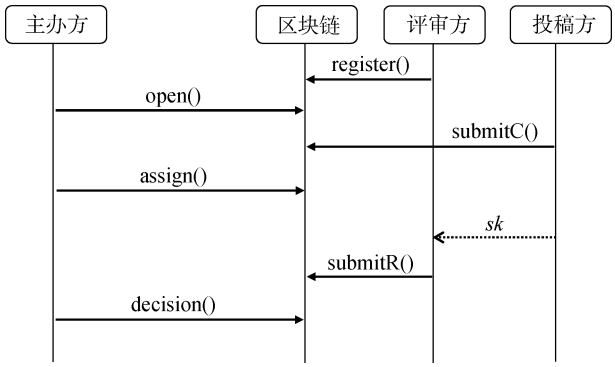

图 3 基础协议概览

Figure 3 Basic protocol sketch

协议具体步骤如下:

(1) 评审方为获得评审资格, 需通过调用函数 register() 进行注册, 该函数需提交评审方个人资料的 *ipfs* 链接, 以允许主办方进行查询和选择。此外, 该函数需向智能合约转账一定数量以太币作为安全押金, 以惩罚评审方可能的恶意行为。评审方仅需注册一次, 使其相关资料关联到智能合约中, 便可参加任意场次论文评审。

(2) 主办方通过调用函数 open() 初始化协议, 并开启提交阶段, 该函数需提交各阶段截止时间与投稿要求的 *ipfs* 链接, 并向智能合约转账规定数量的以太币。

(3) 在提交阶段截止前, 投稿方 C_i 生成一对专用于本次投稿的密钥 $<pk_i^C, sk_i^C>$, 并通过调用函数 submitC() 提交稿件。稿件以稿件对象(creation object, O_i^C)数据结构提交, 具体表示为 $O_i^C := \mathbf{I}(\mathbf{E}(pk_i^C, creation))$, 即稿件内容通过投稿方公钥加密后的 *ipfs* 链接。

(4) 在提交阶段截止后且指派阶段截止前, 主办方通过调用函数 assign() 提交选择的评审方账户地址, 及分配给每个评审方的稿件编号。

(5) 在评审阶段开始, 投稿方通过链下信道向全

部评审方公开其稿件的解密密钥 sk_i^C，从而允许评审方通过 $\mathbf{E}(pk_i^C, creation) \leftarrow \mathbf{R}(O_i^C)$，$creation \leftarrow \mathbf{D}(sk_i^C, \mathbf{E}(pk_i^C, creation))$ 获取稿件内容, 并进行评审。

(6) 在评审阶段截止前, 评审方通过调用函数 submitR() 提交评审结果, 评审结果以打分对象 (score object, O_j^S) 数据结构提交, 具体表示为 $O_j^S := < score, \mathbf{I}(comment) >$, 即稿件评审分值 *score* 及评审意见的 *ipfs* 链接, 其中评审意见为可选项。

(7) 在评审阶段截止后且公示阶段截止前, 主办方通过调用函数 decision(), 基于评审方打分, 在链上计算出各稿件最终分数并进行公示。

4.2 协议改进的主要挑战

可以看到, 在基础协议中, 评审方的身份在指派阶段就已被公开, 且多个步骤要求全部投稿方或评审方调用函数, 导致该协议面临匿名性和可扩展性两个主要挑战。

4.2.1 匿名性

在基础协议中, 评审方通过调用 register() 函数向智能合约进行注册, 这意味着评审方做出承诺, 一旦注册后被某位主办方选择, 将参与对应的评审。在指派阶段, 主办方从注册的评审方中选择一定数量评审方组成评委会, 并通过调用函数 assign() 向智能合约提交评审方与稿件的映射, 这也意味着主办方做出承诺, 在本次评审工作中不改变选择的评审方, 并将给予奖励, 例如将该审稿人的本次服务关联到 publons[26]等学术网站, 帮助该审稿人提升声誉。双方的承诺以函数调用的行为执行, 并以函数执行结果的形式记录进区块链中, 从而被锁定, 不可篡改。在评审完成后, 评审方可在任意时间向智能合约索要奖励, 合约基于承诺记录及评审工作记录, 向合规的评审方给予奖励。然而, 这些记录进区块链的信息是开放、公开的, 这导致评委会信息, 包括被选中的评审者地址及其被分配的稿件, 过早在指派阶段被公开。

基础协议中的评审阶段不具备评审方信息匿名性保障能力, 导致匿名评审转变成投稿方与评审方互相知道对方身份(以太坊账户地址)的明审, 可能引发一系列安全问题。首先, 现有研究工作已证明区块链网络提供的匿名性不强[27], 即从账户地址不难推断出账户拥有者的真实身份。例如, 攻击者可以通过目标账户与其他账户的交互关系, 利用聚类分析的方法, 并结合目标账户及其关联账户在论坛透露的信息, 推测出目标账户地址并发动攻击。其次, 当一个评审方被选择进入某评委会后, 该评审方可能期望向所评审稿件的相关人士收取费用牟利。在这种情况下, 在评审阶段公开的评委会信息有助于该评审方向投稿方证明其身份, 从而促成这一不当交易。因此, 在理想情况下, 区块链不存储任意可能导致指派信息泄露的信息, 包括但不限于被选中评审方的账户地址或其哈希值, 从而使攻击者无法确定性地找出对应某稿件的评审方。

4.2.2 可扩展性

在基础协议中, 智能合约中的函数调用导致了极高 Gas 消耗, 主要存在三个问题:

O(n) 交易数量: 假设参与协议的评审方数量为 n, 且在协议某一步骤要求每个评审方调用智能合约函数, 则协议中将产生不少于 n 个区块链交易。O(n) 的交易量增加了区块链矿工的交易处理压力, 造成性能瓶颈, 可造成区块链网络堵塞。在基础协议中, 投稿方调用函数 submitC() 及评审方调用函数 submitR(), 均导致了 O(n) 交易数量。

O(n) 存储量: 假设协议参与方数量为 n, 且在协议某一步骤中每个参与方向区块链存储大小为 S 的数据, 则协议中将产生不少于 nS 的区块链账本存储量, O(n) 的存储量增加了区块链矿工节点的存储压力, 造成性能瓶颈。同时, 过大的区块链账本易造成区块链矿工节点硬盘空间不足, 导致区块链维护成本提升。在基础协议中, 投稿方调用函数 submitC()、主办方调用函数 assign()、评审方调用函数 submitR()、主办方调用函数 decision() 均产生了 O(n) 存储量。

O(n) 计算量: 假设协议参与方数量为 n, 则在部分场景下(如求和)对其提供的数据的处理需不少于 O(n) 的计算量。在基础协议中, 主办方调用函数 decision() 产生了 O(n) 计算量。

上述问题都会造成区块链系统计算或存储资源的过度消耗, 导致系统表现降低, 极大影响系统工作效率和维护成本, 造成系统瓶颈。更重要的是, 上述问题会造成极高 Gas 消耗, 阻碍协议实用化。为了克服这些问题, 本项目提出轻量级匿名评审协议 BLAR, 该协议轻量化的核心思想是将数据汇聚和处理从区块链链上转移至链下, 而只将处理后的结果在链上做结算, 从而极大降低区块链中交易数量、存储量及计算量。换言之, 轻量级匿名评审协议不依赖区块链进行存储与计算, 而是仅利用区块链进行验证和可信性保证, 从而在最小程度造成区块链链上负担的情况下, 保证数据汇聚与处理的可信性。

4.3 轻量级匿名评审协议

在这一小节, 我们介绍改进后的轻量级匿名评审协议 BLAR。为了方便表述, 我们按照提交、指派、评审、公示四个阶段, 将该协议拆分成四个子协议, 分别在四个小节进行讨论。

4.3.1 方案概述

BLAR 协议的四阶段整体流程如图 4 所示, 其中每个阶段对应一个子协议。论文提交子协议对应投稿方向主办方提交稿件, 匿名指派子协议对应主办方对稿件匿名指派评审方, 论文打分子协议对应评审方提交稿件分数, 结果公示子协议对应主办方公开论文评审结果。在下面四个小节, 我们分别对四个子协议进行详细描述。

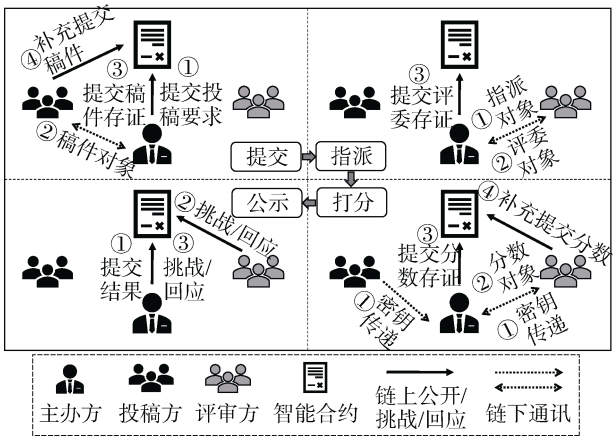

图 4 方案概述图

Figure 4 Scheme overview diagram

4.3.2 论文提交子协议

论文提交子协议概览如图 5 所示。该子协议的参与方包括主办方 H 和 m 位投稿方 $C_{i=1}^{m}$。参与方可通过函数调用交易执行所需函数(图中指向区块链的实线), 也可通过链下信道传递信息(图中点线)。该子协议面临的主要挑战是将基础协议中的 submitC() 函数调用轻量化。

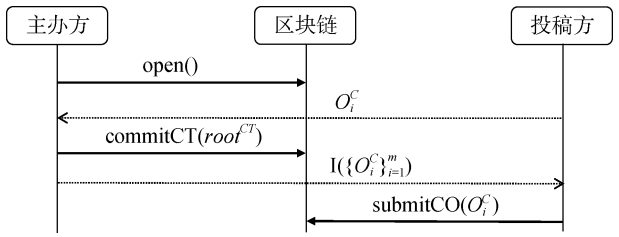

图 5 论文提交子协议

Figure 5 Paper submission sub-protocol

协议具体步骤如下:

(1) 主办方通过调用函数 open() 初始化协议, 并开启提交阶段, 该函数需提交各阶段截止时间与投稿要求的 *ipfs* 链接, 并向智能合约转账一定数量以太币作为安全押金, 目的是惩罚主办方可能的恶意行为。

(2) 在第一子阶段, 投稿方 C_i 形成稿件对象(creation object, O_i^C)数据结构, 具体表示为 $O_i^C := <vrs_i^C, \mathbf{I}(summary, \mathbf{E}(pk_i^C, detail))>$, 其中 $vrs_i^C \leftarrow \mathbf{S}(\mathbf{H}(\mathbf{I}(summary, \mathbf{E}(pk_i^C, detail))))$, 即稿件概要(summary)明文及稿件正文(detail)密文的 *ipfs* 链接与签名。稿件概要包括标题、摘要、关键词等信息, 用以帮助主办方选择合适的审稿方。有别于基础协议中 O_i^C 的链上传输, 本协议为轻量化, 要求投稿方使用链下信道传输 O_i^C。

(3) 在第二子阶段, 主办方收集到来自投稿方的 O_i^C 后, 将 O_i^C 组织成默克尔树 CT, 其中树的叶节点集合 $\{leaf_i^{CT}\}_{i=1}^m \leftarrow \{\mathbf{H}(O_i^C)\}_{i=1}^m$, 树的根节点为叶节点的迭代哈希。主办方通过调用函数 commitCT() 将根节点 $root^{CT}$ 提交到区块链链上; 同时, 主办方将默克尔树 CT 的全部叶节点 $\{leaf_i^{CT}\}_{i=1}^m$ 打包成块(chunk)并经由链下信道向全部投稿方公开。这一策略有以下优点:

- O(1) 链上交易: 只需产生一条交易向链上存储默尔克树根;
- 透明性: 所有稿件叶节点 $\{leaf_i^{CT}\}_{i=1}^m$ 以链下传输形式对全体参与者公开;
- 防篡改: 每个稿件对象 O_i^C 的完整性可通过默克尔根验证;
- 可追溯: 全部稿件对象 O_i^C 对应的投稿方可通过签名追溯。

(4) 在第三子阶段, 我们对主办方可能采取的裁赃这一类恶意行为进行防御。主办方的裁赃恶意行为, 指的是主办方虽然收到了某投稿方 C_i 链下传输的稿件对象 O_i^C, 却故意不将 O_i^C 作为叶子节点包含进默克尔树中, 从而裁赃投稿方未提供稿件。为处理这种可能, 协议在第二子阶段结束后, 留出一小段延长时间, 允许投稿方通过调用函数 submitCO() 上传被主办方恶意屏蔽的稿件对象。该函数具体逻辑如下: 首先进行一次时间条件判断, 强制要求该函数只能在第三子阶段预设时间区间被调用。接着, 存储投稿方提交的 , 并关联主办方。最后, 同时对主

办方和该投稿方记录警告, 以激励双方尽量使用线下方式完成投稿。在这一子阶段, 无论是否之前已通过链下提交投稿, 任何投稿方都可以调用 submitCO() 函数提交投稿, 这能保证任何投稿方都有能力确保其投稿被系统录入, 无论主办方是否可信。然而, 调用 submitCO() 函数需要消耗以太币, 产生花费, 这将激励投稿方优先链下投稿。

4.3.3 匿名指派子协议

匿名指派子协议概览如图 6 所示。该子协议的参与方包括主办方 H 和 n 位评审方 $R_{j=1}^{n}$。该子协议面临两个主要挑战: 一方面需实现匿名性, 即不在链上存储任意可能导致指派信息泄露的信息; 另一方面需将基础协议中的 assign() 函数调用轻量化, 尽可能降低 Gas 消耗。

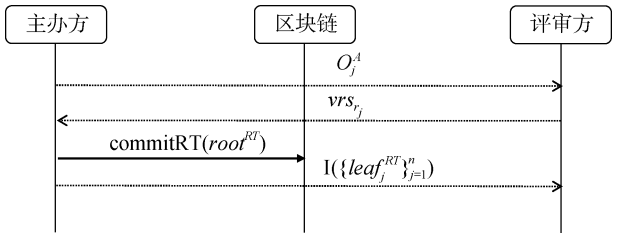

图 6 匿名指派子协议

Figure 6 Anonymous assignment sub-protocol

协议具体步骤如下:

(1) 在第一子阶段, 主办方基于自定义的标准, 从候选评审者集合中选择 n 位评审方 $R_{j=1}^{n}$, 形成本次论文评审的评委会。接着, 主办方对每个评审方分配 k 个待评审稿件, 其中 k 可进行自定义, 如每位审稿人评审 3 篇稿件。

(2) 主办方将全部带签名稿件对象 O_i^C 打包成块(chunk)并经由链下信道向全部评审方公开。此外, 主办方通过链下信道, 向每个评审方单独发送指派对象 $O_j^A =< cn, rid_j, \{cid\}_{i=1}^{k_j} >$, 其中 cn 与 rid_j 分别为评审方 R_j 加入的评委会编号及其在评委会中的序号, $\{cid_i\}_{i=1}^{k_j}$ 为分配给 R_j 的 k_j 个待评审稿件在默克尔树 CT 中对应的叶节点编号。

(3) 在第二子阶段, 评审方 R_j 接收到指派信息 O_j^A 后, 通过链下信道向主办方发送签名信息 $vrs_j^R \leftarrow \mathbf{S}(\mathbf{H}(O_j^A))$, 即评审方 R_j 承诺接收到 O_j^A。

(4) 在第三子阶段, 主办方收集全部 $\{vrs_j^R\}_{j=1}^n$ 后, 构建评委对象 $O_j^R :=< O_j^A, vrs_j^R, vrs_j^{HR} >$, 其中 $vrs_j^{HR} \leftarrow \mathbf{S}(\mathbf{H}(O_j^A, vrs_j^R))$, 即对选择 R_j 的承诺。随后, 主办方将 $\{O^R\}_{j=1}^n$ 组织成默克尔树 RT, 其中叶节点集合 $\{leaf_j^{RT}\}_{j=1}^n \leftarrow \{\mathbf{H}(O_j^R)\}_{j=1}^n$, 根节点为叶节点的迭代哈希 $root^{RT} \leftarrow \mathbf{M}(\{leaf_j^{RT}\}_{j=1}^n)$。主办方通过调用函数 commitRT() 将根节点 R^{RT} 提交到区块链链上; 同时, 主办方将默克尔树 RT 的全部叶节点 $\{leaf_j^{RT}\}_{j=1}^n$ 打包成块(chunk)并经由链下信道向全部评审方公开。

4.3.4 论文打分子协议

论文打分子协议概览如图 7 所示。该子协议的参与方包括主办方 H, m 位投稿方 $C_{i=1}^m$, 以及 n 位评审方 $R_{j=1}^n$。类似论文提交子协议, 论文打分子协议面临的主要挑战是将基础协议中的 submitR() 函数调用轻量化。

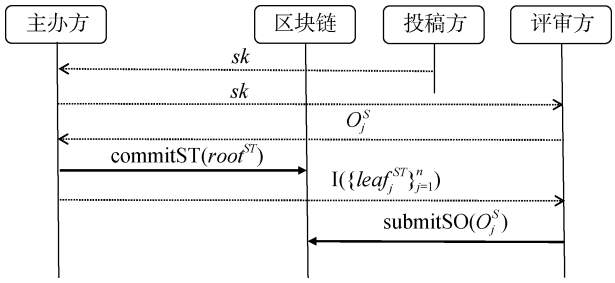

图 7 论文打分子协议

Figure 7 Paper scoring sub-protocol

协议具体步骤如下:

(1) 在第一子阶段, 投稿方通过链下信道向主办方传递其稿件的解密密钥 sk_i^C, 主办方再将 sk_i^C 转发给评审该投稿方稿件的几位评审方, 从而允许评审方通过计算 $\mathbf{E}(pk_i^C, creation) \leftarrow \mathbf{R}(O_i^C)$, $creation \leftarrow \mathbf{D}(sk_i^C, \mathbf{E}(pk_i^C, creation))$ 获取稿件, 并进行评审。

(2) 在第二子阶段, 评审方 R_j 形成分数对象 $O_j^S :=< vrs_j^S, \{s_i\}_{i=1}^{k_j}) >$, 即被分配的 k_j 个待评审稿件的分值及签名 $vrs_j^S \leftarrow \mathbf{S}(\mathbf{H}(\{s_i\}_{i=1}^{k_j}))$。接着, 评审方使用链下信道传输 O_j^S。

(3) 在第三子阶段, 主办方收集到来自评审方的 O_j^S 后, 将 O_j^S 组织成默克尔树 ST, 具体来说, 树的叶节点集合为 $\{leaf_j^{ST}\}_{j=1}^n \leftarrow \{H(O_j^S, vrs_j^{HS})\}_{j=1}^n$, 其中

$vrs_j^{HS} \leftarrow \mathbf{S}(\mathbf{H}(O_j^S))$，树的根节点为叶节点的迭代哈希 $root^{ST} \leftarrow \mathbf{M}(\{leaf_j^{ST}\}_{j=1}^n)$。主办方通过调用函数 commitST() 将根节点 $root^{ST}$ 提交到区块链链上；同时，主办方将全部 $\{leaf_j^{ST}\}_{j=1}^n$ 打包成块(chunk)并经由链下信道向全部评审方公开。

(4) 在第四子阶段，对主办方的栽赃恶意行为进行防御。在此阶段，主办方虽然收到了某评审方 R_j 链下传输的分数对象 O_j^S，却故意不将 O_j^S 作为叶子节点包含进默克尔树 ST 中，从而栽赃评审方未提供分数。为处理这种可能，协议允许评审方通过调用函数 submitSO() 上传被主办方恶意屏蔽的 O_j^S。该函数具体逻辑如下：首先进行一次时间条件判断，强制要求该函数只能在第四子阶段预设时间区间被调用。接着，存储评审方提交的 O_j^S，并关联主办方。最后，同时对主办方和该评审方记录警告，以激励双方尽量遵循前三个子阶段步骤。

4.3.5 结果公示子协议

结果公示子协议概览如图 8 所示。该协议的链上逻辑以包含多个函数的智能合约形式存在。该子协议的参与方包括主办方 H 和 n 位评审方 $R_{j=1}^n$。该子协议的主要功能是对结果及参与方恶意行为进行审计、纠错及追责。该子协议中包含多组有条件触发的函数(图中虚线)，这些函数在协议正常执行时不会被调用，仅在相关参与方存在恶意行为时被调用，用来对恶意行为产生的结果进行纠错，并对恶意行为执行方进行追责。

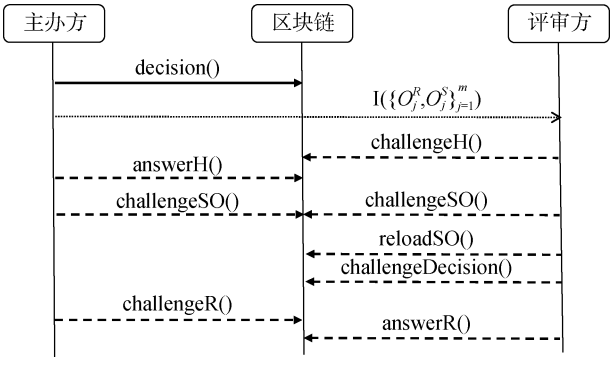

图 8 结果公示子协议

Figure 8 Public notification sub-protocol

协议具体步骤如下：

(1) 在第一子阶段，基于评审方打分，主办方计算出各稿件最终成绩并进行公示。与基础协议不同，为了节省链上成本，计算过程在链下进行，主办方仅通过调用函数 decision() 上传最终结果，如收录论文或获奖论文名单。同时，主办方利用链下信道，公开全部 $\{O_j^R, O_j^S\}_{j=1}^n$。

(2) 在第二子阶段，协议对四类恶意行为进行审计与追责，以保障协议的活性和正确性。

主办方不公开 $<O_j^R, O_j^S>$：在结果公示阶段，如果评审方 R_j 发现主办方链下公开的 $\{O_j^R, O_j^S\}_{j=1}^n$ 不包括自身信息，可通过调用函数 challengeH() 提交 $leaf_j^{RT}$ 与 $leaf_j^{ST}$ 对应的默克尔证明，要求主办方在链上公开 O_j^R 与 O_j^S。该函数首先基于链上默克尔根 $root^{RT}$ 与 $root^{ST}$ 校验默克尔证明，若为真则要求主办方在一定时限内通过调用函数 answerH() 提交 O_j^R 与 O_j^S 到链上，超出时限则合约将主办方标记为违规，没收违规者安全押金并奖励举报者。

评审方 R_j **多次提交** O_j^S：在论文打分子阶段，不诚实的评审方可能利用链上数据与链下数据的割裂，实现上传两次 O_j^S，第一次是通过链下传输，第二次是通过链上调用函数 submitSO()，后者是为了防止主办方的恶意栽赃。没有注意到这一点，两次提交的数据可能会被主办方在链下计算两次，导致错误的链下计算结果。然而，智能合约很难独立监测评审方的这一恶意行为，因为合约不知道链下数据的状态。即使合约知道链下数据的状态，在链上验证该恶意行为也将是非常昂贵的。因此，我们设计 challengeSO() 函数以处理此类错误行为。该函数可以被主办方或任意评审方调用，调用者需提交证据作为函数输入。简单地说，证据是评审方 R_j 已在链下提交 O_j^S 的默克尔证明。该函数首先基于链上默克尔根 $root^{ST}$ 校验默克尔证明，之后校验 R_j 是否通过 submitSO() 提交过 O_j^S。如果两次校验都为真，则合约将 R_j 标记为违规，没收违规者安全押金并奖励举报者。

主办方链下计算结果错误：在结果公示阶段，出于有意或无意，主办方通过函数 decision() 提交的链下计算结果可能是错误的。在这种情况下，任意评审方可以要求将链下计算转换成链上计算，从而保障结果的正确性。具体来说，评审方首先调用函数

reloadSO() 将主办方在论文打分阶段链下公开的 O_j^S 重新提交回链上, 使智能合约能够读取全部 O_j^S。在此基础上, 评审方调用函数 challengeDecision(), 使智能合约利用链上全部 O_j^S 重新计算结果。如果该结果不同于主办方提交的链下计算结果, 则合约将主办方标记为违规, 没收违规者安全押金并奖励举报者。

评审方 R_j 未提交 O_j^S: 在论文打分子阶段, 不诚实的评审方可能没有提交打分 O_j^S。如果这种情况发生, 则主办方或任意评审方可通过调用函数 challengeR() 举报未出现的评审方 R_j。该函数首先校验 R_j 没有在论文打分阶段通过函数 submitSO() 提交 O_j^S 到链上, 如果结果为真, 则要求 R_j 在一定时限内调用函数 answerR() 提交其在链下提交 O_j^S 的默克尔证明, 如果 R_j 超时或 R_j 提交的默克尔证明校验为假, 则合约将 R_j 标记为违规, 惩罚违规者并奖励举报者。

4.3.6 完整 BLAR 协议

完整的轻量级匿名评审协议如图 9 所示, 包含提交、指派、打分、公示四个阶段。

4.3.7 讨论

最后, 我们讨论 BLAR 协议对辩驳(Rebuttal)、不参与分数公开(Opt-out)、补充评审三类功能的支持。

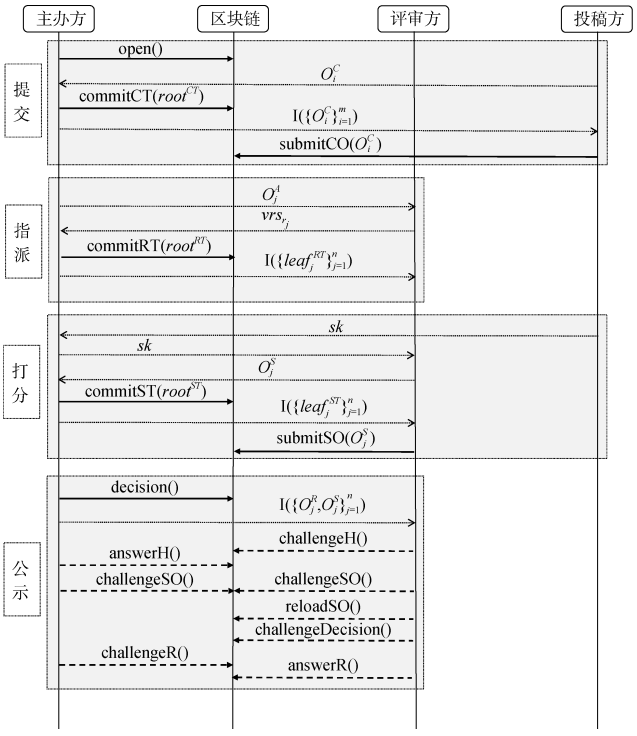

图 9 轻量级匿名评审协议

Figure 9 Lightweight anonymous review protocol

辩驳(Rebuttal): 近年, 越来越多的计算机会议的审稿流程包含辩驳(Rebuttal)阶段, 允许投稿方基于评审意见提交一定字数的 Rebuttal, 且允许评审方在阅读 Rebuttal 内容后修改分数。通过在现有的打分阶段和公示阶段之间, 新增辩驳阶段和第二次打分阶段, BLAR 协议能够支持这类功能。首先, 在打分阶段后, 新增辩驳阶段。该阶段中, 主办方将分数及评审意见 IPFS 链接通过链下信道告知投稿方, 投稿方在辩驳阶段截止前, 可选择向智能合约提交 Rebuttal 内容 IPFS 链接, 其具体方法类似将现有提交阶段中提交的稿件内容替换为 Rebuttal 内容。接着, 在辩驳阶段和公示阶段间, 新增一轮打分阶段。该阶段中, 类似 Rebuttal 前的首次打分, 评审方可基于 Rebuttal 内容进行重新打分。

不参与分数公开(Opt-out): 被拒的投稿方可能不希望其分数被公开, 即可能希望选择不参与分数公开(opt-out)。通过简单的修改, BLAR 协议能够支持这类功能。首先, 在提交阶段, 投稿方在提交稿件时标注 opt-out, 并生成一对投稿专用非对称密钥, 公开其中公钥, 通过链下信道将私钥告知主办方。接着, 在打分阶段, 如评审方发现其被分配的稿件标注有 opt-out, 需先使用该稿件对应公钥对其分数进行加密, 再基于加密后的分数生成分数对象。最后, 在公示阶段, 主办方能够对所有分数进行解密, 以进行录用决策。同时, 主办方公开全部分数对象, 但其中标注 opt-out 的稿件分数, 仅能被该稿件的投稿方解密获悉。

补充评审: 如果打分阶段部分论文收到的意见数量不足, 可在公示阶段前重复进行一轮“指派-打分”补充论文意见, 再对结果进行公开。

5 安全与成本分析

在本节中, 我们首先对本文提出的轻量级匿名评审协议 BLAR 进行安全性分析。接着, 我们分析该协议的链上执行成本。

5.1 安全性分析

定理 1.本文提出的轻量化匿名评审协议 BLAR 是符合定义 1 的安全匿名评审协议.

安全匿名评审协议需满足活性、正确性、匿名性三类安全特性, 我们依次进行证明。

引理 1. BLAR 协议具备活性.

证明. 我们首先假设仅有主办方是诚实参与者, 则主办方有能力在前三阶段顺序执行后的结果公示阶段通过调用函数 decision() 产生评审结果。接着, 我们假设仅有一位评审方是诚实参与者, 则该评审

方有能力在结果公示阶段通过调用函数 reloadSO() 将全部 O_j^S 提交到链上，并通过调用函数 challengeDecision() 在链上计算评审结果。综上，只要主办方和评审方存在至少一位诚实参与者，必可通过函数 decision() 或 challengeDecision() 使协议执行完毕且产生评审结果，故 BLAR 协议具备活性.

引理 2. BLAR 协议具备正确性.

证明. 我们首先假设仅有主办方是诚实参与者，则主办方有能力在结果公示阶段基于链上和链下的打分情况 O_j^S 计算出正确结果，并经由函数 decision() 提交到链上确认。接着，我们假设仅有一位评审方是诚实参与者，则不诚实的主办方可能利用函数 decision() 提交错误链下计算结果，甚至不提交结果。在这些情况下，该评审方有能力在结果公示阶段通过调用函数 reloadSO() 将全部 O_j^S 提交到链上，并通过调用函数 challengeDecision() 在链上计算出于基于全部 O_j^S 的正确结果。综上，只要主办方和评审方存在至少一位诚实参与者，必可通过函数 decision() 或 challengeDecision() 在链上产生符合评审者打分的正确结果，故轻量化匿名评审协议 BLAR 具备正确性.

引理 3. BLAR 协议具备匿名性.

证明. 我们假设主办方在公示期前不透露指派信息，统计评审方在各阶段可获得的评审相关信息。在提交阶段，没有产生评审相关信息。在指派阶段，评审方可获得默克尔树 RT 的根节点 $root^{RT}$ 和全部叶节点 $\{leaf_j^{RT}\}_{j=1}^n$。在打分阶段，评审方可获得树 ST 的根节点 $root^{ST}$ 和叶节点 $\{leaf_j^{ST}\}_{j=1}^n$。其中，叶子节点 $leaf_j^{RT} \leftarrow \mathbf{H}(< O_j^A, vrs_j^R, vrs_j^{HR} >)$，$leaf_j^{ST} \leftarrow \mathbf{H}(< \{s\}_{i=1}^k, vrs_j^S, vrs_j^{HS} >)$，且评审方不知道 vrs_j^{HR} 与 vrs_j^{HS}，只要哈希算法 $\mathbf{H}(\bullet)$ 与签名算法 $\mathbf{S}(\bullet)$ 是安全的，则评审方无法证明已被主办方选择进入评委会。以太坊使用的 Keccak-256 哈希算法与 ECDSA 数字签名算法的安全性已在相关研究中得到证明[17-18]，故本文提出的轻量化匿名评审协议 BLAR 具备匿名性。

5.2 成本分析

定理 2.本文提出的轻量化匿名评审协议是符合定义 2 的可扩展匿名评审协议.

证明. 我们假设全部协议参与方具备理性，在此基础上，对匿名评审协议的四个子协议的成本进行逐一分析，从而获得执行完整协议的成本，以论证协议的高效性和可扩展性。

在论文提交子协议中，若仅需调用函数 commitCT()，则 Gas 消耗为 O(1)；若投稿方需通过调用函数 submitCO() 上传被主办方恶意屏蔽的 O_i^C，则 Gas 消耗将增加，上限为 O(n)；由于 submitCO() 的调用不仅能够修正主办方恶意行为的危害，还将导致主办方与调用该函数的投稿方共同支付交易费用，受到经济惩罚，产生负收益，理性的主办方与投稿方不会使函数 submitCO() 被调用，故该子协议仅需调用函数 commitCT()，从而使 Gas 消耗为 O(1)。

同理，在匿名指派子协议与论文打分子协议，理性的参与方仅需调用函数 commitRT() 与 commitST()，从而使 Gas 消耗为 O(1)。

在结果公示子协议中，若仅需调用函数 decision()，则 Gas 消耗为 O(1)；若存在四类恶意行为，需调用 challengeH()、challengeSO()、challengeDecision()、challengeR() 等函数，则 Gas 消耗将增加，上限为 O(n)；由于这些函数的调用不仅能够修正恶意行为的危害，还将导致恶意行为执行方安全押金被没收，受到经济惩罚，产生负收益，理性的协议参与方不会使这些函数被调用，故该子协议仅需调用函数 decision()，从而使 Gas 消耗为 O(1)。

综上，当全部协议参与方具备理性，协议执行过程仅需调用函数 commitCT()、commitRT()、commitST()、decision()，Gas 消耗为 O(1)，故轻量化匿名评审协议是可扩展匿名评审协议。

6 实验与评估

在本节中，我们进行了一系列实验，对本文提出的轻量级匿名评审协议 BLAR 进行评估。我们采用 Solidity 语言[13]编写了智能合约，作为协议在区块链链上部分的执行逻辑。我们采用以太坊官方测试网络 kovan[28]对协议进行测试评估。与近期相关工作[23,29]一样，本文主要关注协议执行过程中的链上效率，即协议在以太坊环境中执行消耗的 Gas 量，该指标代表协议的链上执行复杂度与成本，此外，我们还将提出的轻量级匿名评审协议与包含 Ants-Review 协议在内的多个相关工作中提出的协议进行对比。

6.1 Gas 消耗量

在表 2 中，我们面向本文提出的轻量级匿名评审

协议 BLAR, 列出了不同阶段可供调用的智能合约函数, 这些函数的调用者与函数功能, 以及函数调用消耗的 Gas。BLAR 协议的 Gas 消耗具体包括以下部分:

(1) 在论文提交阶段, 主办方需消耗 69006 Gas 调用函数 commitCT() 提交默克尔树 ct 的根节点 R^{ct}, 从而避免上传全部稿件对象 O_i^C。同时, 协议允许投稿方 C_i 消耗 137258 Gas 调用函数 submitCO() 提交 O_i^C, 以防止主办方恶意屏蔽其投稿。在这一阶段, commitCT() 为协议中必须调用的函数, submitCO() 仅为抵御主办方恶意行为而设计, 在参与方理性条件下无需被调用, 为非必须调用的函数, 因此, 在参与方理性条件下, 这一阶段的 Gas 消耗总量为 69006。

(2) 在匿名指派阶段, 主办方需消耗 67317 Gas 调用函数 commitRT() 提交默克尔树 RT 的根节点 $root^{RT}$, 从而避免上传评审对象 O_j^R, 在保障匿名性的同时, 降低链上消耗。在这一阶段, commitCT() 为必须调用的函数, 无其他智能合约函数, 因此这一阶段的 Gas 消耗总量为 67317。

(3) 在论文打分阶段, 主办方需消耗 66429 Gas 调用函数 commitST() 提交默克尔树 ST 的根节点 $root^{ST}$, 从而避免上传全部打分对象 O_j^S。同时, 为便于测试, 我们假设每个评审方需对三篇稿件进行打分, 且分数为区间 [0,10] 的任意整数。协议允许评审方 R_j 消耗 145366 Gas 调用函数 submitSO() 提交 O_j^S, 以防止主办方恶意屏蔽其打分。在这一阶段, commitST() 为必须调用的函数, submitSO() 仅为抵御主办方恶意行为而设计, 在参与方理性条件下无需使用, 为非必须调用的函数, 因此这一阶段的 Gas 消耗总量为 66429。

(4) 在结果公开阶段, 我们假设主办方计划根据每篇稿件获得的评价分对稿件进行排序, 并选取前三名稿件作为优胜者。在现实应用中, 此阶段优胜者可对应竞赛晋级名单, 或论文录用名单等。基于这一假设, 主办方需消耗 60233 Gas 调用函数 decision() 提交链下计算执行结果, 即三名优胜者的名单。在这一阶段, 我们设计了一系列恶意行为的抵御方法: 其一, 为抵御主办方不公开 $<O_j^R, O_j^S>$, 协议允许评审方 R_j 消耗 61578 Gas 调用函数 challengeH(), 迫使主办方消耗 59318 Gas 调用函数 answerH() 将 $<O_j^R, O_j^S>$ 提交至链上; 其二, 为抵御评审方 R_j 多次提交 O_j^S, 协议允许主办方或评审方消耗 66365 Gas 调用函数 challengeSO(), 使智能合约基于链上记录审计并惩罚该恶意行为; 其三, 为抵御主办方链下计算结果错误, 协议允许评审方消耗 $(38147+36943N^{SO})$ Gas 调用函数 reloadSO() 将一批 N^{SO} 个 O_j^S 从链下提交至链上(reloadSO() 函数允许调用者批量上传 O_j^S, 这比逐一上传 O_j^S 更加节省 Gas), 在通过多次调用 reloadSO() 将全部 O_j^S 提交至链上后, 评审方消耗 $(48231+2373N^C)$ Gas 重新在链上计算优胜者名单, 其中 N^{SO} 与 N^C 分别代表一个 O_j^S 数据块包含的 O_j^S 数量及参与协议的投稿方数量; 最后, 为抵御评审方 R_j 不提交 O_j^S, 协议允许主办方消耗 82664 Gas 调用函数 challengeR(), 迫使 R_j 消耗 30144 Gas 调用函数 answerR() 证明其打分 O_j^S 已提交。在这一阶段, 尽管可供调用的函数众多, 仅 decision() 为必须调用的函数, 其他所有函数均为抵御相关参与方恶意行为而设计, 在参与方理性条件下无需使用, 为非必须调用的函数, 因此这一阶段的 Gas 消耗总量为 60233。

基于上述实验结果与分析, 通过统计四阶段必须调用函数对应的 Gas 消耗, 本文提出的轻量级匿名评审协议的 Gas 消耗总量仅为 262985。更重要的是, 该 Gas 消耗独立于协议参与方的规模, 具备可扩展性。此外, 为了更直观反映协议实际消耗的金钱量, 我们从 Etherscan 网站[30]获取了 2019 年上半年以太币兑换 Gas 的均值汇率, 以及美元兑换以太币的均值汇率, 分别为 1.67×10^{-8} 及 175, 故 BLAR 协议成本为 $262985\times1.67\times10^{-8}\times175\approx0.77$ 美元。

6.2 与现有工作对比

在这一小节, 我们将本文提出的轻量级匿名评审协议 BLAR 与在 4.1 小节描述的基础协议, 文献[12]提出的 Ants-Review 协议, 以及文献[31]提出的 EthReview 协议进行对比。其中, Ants-Review 协议是一种基于以太坊的同行评审协议, 与本文场景完全一致, 我们已在引言部分对该协议进行了介绍。EthReview 协议是一种基于以太坊的产品评论协议, 其场景类似网上购物后买方给卖方打分, 与本文场景存在相似性, 因此被纳入对比。接下来, 我们首先在表 3 中从匿名性和 Gas 消耗两方面对上述四个协议进行对比, 并进一步通过图 10 对比上述四个协议的执行成本。

表 2 协议中各阶段主要函数及其 Gas 消耗

Table 2 The key functions and Gas consumption in each phase of the protocol

阶段	函数	调用者	功能	Gas 消耗
论文提交	commitCT()	主办方	提交默克尔根 $root^{CT}$	69006
	submitCO()	投稿方	提交稿件对象 O_i^C	137258
匿名指派	commitRT()	主办方	提交默克尔根 $root^{RT}$	67317
论文打分	commitST()	主办方	提交默克尔根 $root^{ST}$	66429
	submitSO()	评审方	提交分数对象 O_j^S	145366
	decision()	主办方	提交链下计算结果	60233
	challengeH()	评审方	质疑主办方不公开 $<O_j^R, O_j^S>$	61578
	answerH()	主办方	响应 challengeH() 质疑	59318
结果公示	challengeSO()	主办方或评审方	质疑评审方 R_j 多次提交 O_j^S	66365
	reloadSO()	评审方	链下公开的 O_j^S 重新提交回链上	38147+36943 N^{SO}
	challengeDecision()	评审方	质疑链下计算结果	48231+2373 N^C
	challengeR()	主办方	质疑评审方 R_j 未提交 O_j^S	82664
	answerR()	评审方	响应 challengeR() 质疑	30144

表 3 现有工作对比

Table 3 Comparison with existing works

	匿名性	Gas 消耗
Ants-Review 协议	×	$O(n)$
基础协议	×	$O(n)$
EthReview 协议	×	$O(n)$
BLAR 协议	√	$O(1)$

文献[12]提出的 Ants-Review 协议目前不支持匿名性, 且由于每个评审方提交评审意见需调用多个智能合约函数, 产生了 O(n) 级别的 Gas 消耗。在 4.1 小节描述的基础协议也不支持匿名性, 且由于每个投稿方需通过调用函数 submitC() 提交稿件, 且每个评审方需通过调用函数 submitR() 提交评审结果, 产生了 O(n) 级别的 Gas 消耗。文献[31]提出的 EthReview 仅利用以太坊地址充当用户假名, 无法支持匿名性, 且产品评分记录进区块链账本需经由买方提交及背书人背书, 调用了多个智能合约函数, 产生了 O(n) 级别的 Gas 消耗。综上, 本文提出的轻量级匿名评审协议是首个兼具匿名性及可扩展性的去中心化评审协议。

接着, 在图 10 中, 为了便于直观理解, 我们基于 1.67×10^{-8} 以太币/Gas、175 美元/以太币的汇率, 将执行上述协议的 Gas 总消耗量换算成美元。此外, 为了对比和评估这些协议的可扩展性, 我们将协议参与者的规模 n 从 10 指数增加到 1000。可以看到, 随着 n 的指数增加, Ants-Review 协议、基础协议、EthReview 协议的执行成本均随之迅速增加, 分别从 11.4 美元增加到 357.5 美元, 从 4.6 美元增加到 397.1 美元, 从 4.5 美元增加到 445.4 美元。与之相比, 本文提出的 BLAR 协议的执行成本没有增加, 保持在 0.77 美元, 从而证明了 BLAR 协议的可扩展性。

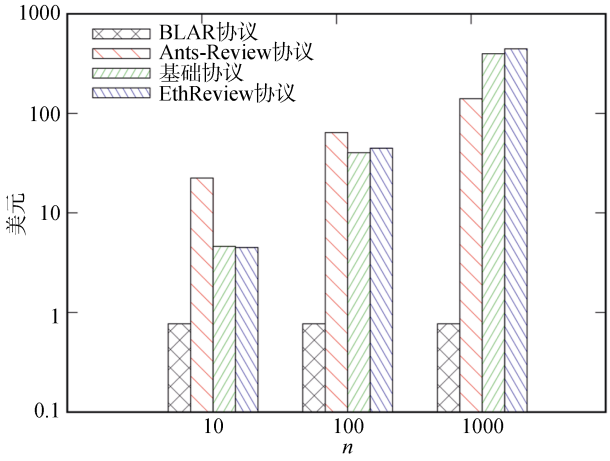

图 10 协议执行成本(美元)对比

Figure 10 Comparison of USD cost of the protocols

7 结论

本文提出一种基于区块链的轻量级匿名审稿协议 BLAR, 解决了去中心化评审协议的匿名性和可扩展性两个关键问题。BLAR 协议不需要在评审结果展示前在区块链上存储任意可能导致指派信息泄露的信息, 从而使攻击者无法确定性地找出对应某稿件的评审方。同时, BLAR 协议不依赖区块链进行存储与计算, 从而在最小程度造成区块链链上负担的

情况下, 保证数据汇聚与处理的可信性。我们证明, 只要主办方和评审方存在至少一位诚实参与者, BLAR 协议一定能执行完毕, 且一定能产生符合评审者打分的正确结果。我们还证明, 当全部协议参与方具备理性时, 执行 BLAR 协议全过程的成本为 O(1), 独立于参与方数量的规模。我们在以太坊官方测试网络上实现了 BLAR 协议, 并进行了实验评估。结果表明, 无论协议参与方的规模大小, BLAR 协议都可以匿名评审的执行成本降低到 1 美元以下, 远低于现有相关工作中协议的执行成本, 具备了实用性。在未来的工作中, 我们将重点关注恶意主办方提前透露指派信息场景下的评审匿名性保障方法。

参考文献

[1] Tennant J P, Dugan J M, Graziotin D, et al. A Multi-Disciplinary Perspective on Emergent and Future Innovations in Peer Review[J]. *F1000Research*, 2017, 6: 1151.

[2] Ross-Hellauer T. What is Open Peer Review? a Systematic Review[J]. *F1000Research*, 2017, 6: 588.

[3] Nature. Nature will Publish Peer Review Reports as a Trial[J]. *Nature*, 2020, 578(7793): 8.

[4] Tran D, Valtchanov A, Ganapathy K, et al. An Open Review of OpenReview: A Critical Analysis of the Machine Learning Conference Review Process[EB/OL]. 2020: *ArXiv Preprint* ArXiv: 2010.05137.

[5] Chen J, Yao S X, Yuan Q, et al. CertChain: public and efficient certificate audit based on blockchain for TLS connections[C]. *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018: 2060-2068.

[6] Hu S S, Cai C J, Wang Q, et al. Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization[C]. *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018: 792-800.

[7] Wright C. Bitcoin: A Peer-to-Peer Electronic Cash System[J]. *SSRN Electronic Journal*, 2008: 21260.

[8] Buterin V. A Next-Generation Smart Contract and Decentralized Application Platform[J]. *White Paper*, 2014: 3(37).

[9] Wood G. Ethereum: A Secure Decentralised Generalised Transaction Ledger[J]. *Ethereum Project Yellow Paper*, 2014: 1-32.

[10] Ethereum market cap. https://coinmarketcap.com/coins/. Aug. 2021.

[11] State of the dapps. https://www.stateofthedapps.com/. Aug. 2021.

[12] Trovò B, Massari N. Ants-Review: A Privacy-Oriented Protocol for Incentivized Open Peer Reviews on Ethereum[M]. Euro-Par 2020: Parallel Processing Workshops. Cham: Springer International Publishing, 2021: 18-29.

[13] The solidity contract-oriented programming language. https://github. com/ethereum/solidity. Aug. 2021.

[14] Gervais A, Karame G O, Wüst K, et al. On the Security and Performance of Proof of Work Blockchains[C]. *The 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016: 3-16.

[15] Whisper protocol. https://github.com/ethereum/wiki/wiki/Whisper. Aug. 2021.

[16] Merkle R C. A Digital Signature Based on a Conventional Encryption Function[C]. *CRYPTO '87: A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology*, 1987: 369-378.

[17] Bertoni G, Daemen J, Peeters M, et al. Keccak[M]. Advances in Cryptology - EUROCRYPT 2013. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013: 313-314.

[18] Johnson D, Menezes A, Vanstone S. The Elliptic Curve Digital Signature Algorithm (ECDSA)[J]. *International Journal of Information Security*, 2001, 1(1): 36-63.

[19] Benet J. IPFS - Content Addressed, Versioned, P2P File System[EB/OL]. 2014: arXiv: 1407.3561. https://arxiv.org/abs/1407.3561

[20] Maram S K D, Zhang F, Wang L, et al. CHURP: Dynamic-Committee Proactive Secret Sharing[C]. *The 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019: 2369-2386.

[21] Kosba A, Miller A, Shi E, et al. Hawk: the blockchain model of cryptography and privacy-preserving smart contracts[C]. *2016 IEEE Symposium on Security and Privacy*, 2016: 839-858.

[22] Dong C Y, Wang Y L, Aldweesh A, et al. Betrayal, Distrust, and Rationality: Smart Counter-Collusion Contracts for Verifiable Cloud Computing[C]. *The 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017: 211-227.

[23] Li C, Palanisamy B, Xu R H, et al. NF-crowd: Nearly-free blockchain-based crowdsourcing[C]. *2020 International Symposium on Reliable Distributed Systems*, 2020: 41-50.

[24] Kalodner H, Goldfeder S, Chen X Q, et al. Arbitrum: Scalable, Private Smart Contracts[C]. *The 27th USENIX Conference on Security Symposium*, 2018: 1353-1370.

[25] Zhang F, He W, Cheng R, et al. The ekiden platform for confidentiality-preserving, trustworthy, and performant smart contracts[C]. *IEEE Security & Privacy*, 2019: 17-27.

[26] Publons. http://publons.com. Aug. 2021.

[27] Biryukov A, Khovratovich D, Pustogarov I. Deanonymisation of Clients in Bitcoin P2P Network[C]. *The 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014: 15-29.

[28] Kovan. https://kovan.etherscan.io/. Aug. 2021.

[29] Dziembowski S, Eckey L, Faust S, et al. Perun: virtual payment hubs over cryptocurrencies[C]. *2019 IEEE Symposium on Security and Privacy*, 2019: 106-123.

[30] Etherscan: gas price. https://etherscan.io/chart/gasprice. Aug. 2021.

[31] Zulfiqar M, Tariq F, Janjua M U, et al. EthReview: An Ethereum-Based Product Review System for Mitigating Rating Frauds[J]. *Computers & Security*, 2021, 100: 102094.

李超 于 2019 年在匹兹堡大学信息科学专业获得博士学位。现任北京交通大学计算机学院讲师。研究领域为网络空间安全。研究兴趣包括: 区块链、隐私计算。Email: li.chao@bjtu.edu.cn

王健 于 2008 年在北京邮电大学获得博士学位。现任北京交通大学计算机学院副教授。研究领域为网络空间安全。研究兴趣包括: 密码应用及区块链、网络安全。Email: wangjian@bjtu.edu.cn

刘吉强 于 1999 年在北京师范大学获得理学博士学位。现任北京交通大学计算机学院教授。研究领域为网络空间安全。研究兴趣包括: 可信计算、隐私保护、物联网安全。Email: jqliu@bjtu.edu.cn